Final Report: AI-Based Traditional Checkers Game

Submitted By:

• M. Hassan Khan (22k-4348)

• Unzila Javed (22k-4168)

Course: Artificial Intelligence

Instructor: Alina Arshad / Almas Ayesha Ansari

Submission Date: 5/8/25

1. Project Overview

Project Topic:

This project focuses on developing an AI-based traditional Checkers game where a human player competes against an AI opponent. The game adheres to the standard Checkers rules and introduces intelligent decision-making through AI.

Objective:

• Develop a strategic AI opponent for Checkers that can challenge human players using intelligent search algorithms.

• Implement the Minimax algorithm with Alpha-Beta Pruning for efficient decision-making.

• Explore AI search techniques, heuristic design, and optimization strategies to simulate human-like gameplay.

2. Game Description

Original Game Background:

Checkers is a classic two-player board game played on an 8x8 board using 12 pieces per player placed on dark squares. Players take turns moving pieces diagonally and capturing opponent pieces by jumping over them. Pieces that reach the last row are promoted to 'Kings' with enhanced movement.

Innovations Introduced:

- • AI Opponent using Minimax Algorithm enhanced with Alpha-Beta Pruning.
- • Visualization of AI move evaluations for better player understanding.

- • Adaptive Difficulty levels based on AI's search depth.
- • AI Move Explanations (optional) to educate players on AI strategies.
- • Custom Rule Innovation: If one piece captures two opponent pieces in a single move, it is instantly promoted to a King and gains multidirectional movement (forward, backward, diagonal in all directions).

3. AI Approach and Methodology

AI Techniques Used:

- • Minimax Algorithm: To simulate all possible moves and select optimal strategies.
- • Alpha-Beta Pruning: Optimizes the Minimax process by pruning unnecessary branches.
- • Heuristic Evaluation Function: Designed to assess board states effectively.
- • Optional Reinforcement Learning: For future AI self-improvement across games.

Heuristic Design:

- • Number of remaining pieces.
- • King piece advantage.
- • Control over the center of the board.
- • Detection of forced captures and double-jump opportunities.

Complexity Analysis:

• Minimax Time Complexity: O(b^d), where b is the branching factor and d is depth.

• Alpha-Beta Pruning significantly reduces the number of nodes evaluated, enabling deeper and more efficient analysis.

4. Game Rules and Mechanics

Modified Rules:

The game mainly follows traditional Checkers rules with the addition of a special King promotion rule when two opponent pieces are captured in a single move.

Winning Conditions:

• A player wins by capturing all opponent pieces or blocking them from making any legal moves.

Turn Sequence:

- • Players alternate turns.
- • Capturing is mandatory if available.
- • Kings can move and capture in all directions.

5. Implementation Plan

Programming Language:

Python

Libraries and Tools:

- • Pygame: For GUI development and game rendering.
- • NumPy: For handling game state data and computations.
- • TensorFlow/Keras (optional): For reinforcement learning enhancements.

Milestones and Timeline:

- • Week 1–2: Develop game logic, board layout, and enforce rules.
- • Week 3–4: Implement AI (Minimax with Alpha-Beta Pruning, heuristics).
- • Week 5–6: Design GUI and improve AI performance.
- • Week 7: Introduce difficulty levels and final testing.
- • Week 8: Complete documentation and submit final project.

6. References

- • Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction.
- • Russell, S. J., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach.
- • Online research articles and documentation related to Checkers AI and heuristic development.